Claims

1. (Original) One or more computer-readable media with computer-executable instructions for implementing a software development architecture comprising:

a software development scenario-independent intermediate representation format;

one or more exception handling models operable to support a plurality of programming language specific exception handling models;

a type system operable to represent the type representations of a plurality of source languages; and

a code generator operable to generate code targeted for a plurality of execution architectures.

2. (Original) The one or more computer-readable media of claim 1 wherein the architecture is scalable to produce target software development tools ranging from lightweight JIT compilers to whole program optimizing compilers.

3. (Original) The one or more computer-readable media of claim 1 wherein the architecture can be configured to produce a target software development tool with varying ranges of memory footprint, compilation speed, and optimization.

4. (Original) The one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool modifiable by combining a modification component with the software development architecture.

5. (Original) The one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool by dynamically linking a binary version of the software development architecture to a modification component.

6. (Original) The one or more computer-readable media of claim 1 wherein the intermediate representation format is extensible at runtime of a software tool employing the intermediate representation format.

7. (Original) The one or more computer-readable media of claim 1 wherein the architecture is combinable with one or more software development components.

8. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components comprise data describing a target software development tool.

9. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components provides target execution architecture data to the code generator.

10. (Original) The one or more computer-readable media of claim 7 wherein the one or more software development components provide one or more type-checking rules to the type system.

11. (Currently Amended) The one or more computer-readable media of claim 7 wherein the one or more software development components provide a set of class extension declarations to the architecture.

12. (Original) The one or more computer-readable media of claim 7 wherein the combined one or more software development components and architecture produce a target software development tool.

13. (Original) The one or more computer-readable media of claim 12 wherein the target software development tool comprises a native compiler.

14. (Original) The one or more computer-readable media of claim 12 wherein the target software development tool comprises a JIT compiler.

15. (Original) A method of creating a target software development tool, the method comprising:

receiving at least one computer-readable specification specifying functionality specific to one or more software development scenarios;

creating at least one software development component from the at least one specification; and

integrating the at least one software development component into a software development scenario-independent framework.

16. (Original) The method of claim 15 further comprising:

compiling the at least one software development component and framework to create the target software development tool.

17. (Original) The method of claim 15 wherein software development components created from a plurality of computer-readable specifications for a plurality of respective software development scenarios are integrated into the framework.

18. (Original) The method of claim 17 wherein the plurality of computer-readable specifications specify functionality for the following respective software development scenarios:

target execution architecture;

input language or input binary format; and

compilation type.

19. (Original) The method of claim 15 wherein the computer-readable specification specifies functionality for a target execution architecture of the software development tool.

20. (Original) The method of claim 15 wherein the computer-readable specification specifies functionality for accommodating an input language for the software development tool.

21. (Original) The method of claim 15 wherein the computer-readable specification specifies functionality for accommodating a binary input for the software development tool.

22. (Original) The method of claim 15 wherein the computer-readable specification comprises one or more rulesets for type-checking one or more languages.

23. (Original) The method of claim 15 wherein the computer-readable specification comprises a set of class extension declarations specific to one or more of the software development scenarios.

24. (Original) The method of claim 15 wherein the computer-readable specification comprises functionality for processing an intermediate representation format capable of representing a plurality of programming languages.

25. (Original) The method of claim 24 wherein the intermediate representation format comprises one or more exception handling models capable of supporting a plurality of programming language-specific exception handling models.

26. (Original) The method of claim 24 wherein the intermediate representation comprises type representations capable of representing the type representations of a plurality of programming languages.

27. (Original) The method of claim 15 further comprising:
integrating custom code specific to one of the software development scenarios.

28. (Original) The method of claim 15 wherein the software development tool comprises one of the group consisting of: a native compiler, a JIT compiler, an analysis tool, and a CDK.

29. (Original) The method of claim 15 wherein the computer-readable specification specifies functionality of one of the group consisting of: a Pre-JIT compiler functionality, optimizer functionality, and defect detection tool functionality.

30. (Original) One or more computer-readable media containing one or more computer-executable instructions for performing the method of claim 15.

31. (Original) A method of creating a target software development tool from a common framework, the method comprising:

configuring the common framework based on one or more characteristics of the target software development tool;

integrating data comprising one or more characteristics of the target software development tool into the common framework; and

creating the target software development tool from the integrated common framework.

32. (Original) The method of claim 31 wherein the one or more characteristics can comprise the amount of memory necessary for the target software development tool to execute on a target architecture, the speed at which the target software development tool will execute on a target architecture, an input language for the target software development tool, a input binary format for the target software development tool, or the target architecture for the target software development tool to execute on a target architecture.

33. (Original) A method of creating a software development tool, the method comprising:

receiving at least one computer-readable specification specifying:

a type of software development tool to be created,

functionality for a target execution architecture for the software development tool,

functionality for accommodating an input language for the software development tool, and

functionality for processing an intermediate representation capable of representing a plurality of programming languages,

wherein the at least one computer-readable specifications comprises a set of class extension declarations;

integrating the at least one computer-readable specification into a software development architecture; and

creating the software development tool via the integrated specification and architecture, wherein the software development tool is of the type specified, is targeted to the target execution architecture specified, accommodates the input language specified, and processes the intermediate representation.

34. (Original) A method of producing inter-compatible software development tools, the method comprising:

creating a first software development tool from a software development architecture; and

creating a second software development tool based on the first software development tool, wherein the second software development tool dynamically links to a binary version of the software development architecture.

35. (Original) The method of claim 34 wherein the binary version of the software development architecture contains classes that are extensible through a set of declarations.

36. (Original) The method of claim 34 wherein the software development architecture comprises functionality for an intermediate representation format used by both the first and second software development tools.

37. (Original) The method of claim 34 wherein the software development architecture comprises functionality for a type system used by both the first and second software development tools.

38. (Original) The method of claim 34 wherein the software development architecture comprises functionality for exception handling models used by both the first and second software development tools.

39. (Original) A method of modifying a software development tool, the software development tool having been created using a software development architecture comprising one or more software development components, the method comprising:

dynamically linking a software development component not present in the software development architecture to a binary version of the software development architecture; and

creating a modified software development tool from the dynamically linked binary version and the software development component.

40. (Original) The method of claim 39 wherein the binary version of the software development architecture comprises classes that are extensible through a set of declarations.

41. (Original) The method of claim 39 wherein the binary version of the software development architecture comprises functionality for a type system used by the modified software development tool.

42. (Original) The method of claim 39 wherein the binary version of the software development architecture comprises functionality for exception handling models used by the modified software development tool.

43. (Currently Amended) A method of creating a software development tool, the method comprising:

receiving at least one computer-executable file comprising:

an intermediate representation capable of representing a plurality of programming languages and computer executable images;

one or more exception handling models capable of supporting a plurality of programming language specific exception handling models;

a type system capable of representing the type representations of a plurality of source languages; and

a code generator capable of generating code targeted for a plurality of execution architectures;

linking a software component to the at least one computer-executable file using at least one class extension declaration[[s]]; and

creating the software development tool via the linked software component and computer-executable file.